New Astronomy 65 (2018) 59–66

Contents lists available at ScienceDirect

New Astronomy

journal homepage: www.elsevier.com/locate/newast

DeltaComp: Fast and efficient compression of astronomical timelines

Sebastian Deorowicz*[,a], Szymon Grabowski[b]

[a] Institute of Informatics, Silesian University of Technology, Gliwice, Poland
[b] Institute of Applied Computer Science, Lodz University of Technology, Łódź, Poland

ARTICLE INFO

Keywords:
Information systems
Data compression
Time-series data
Numerical methods

ABSTRACT

Astronomical instruments commonly generate large data series in tabular format. To efficiently store and transmit these data series, carefully designed compression formats are welcome. We propose DeltaComp, a free open-source program suited for (relatively) smooth streams of data. DeltaComp is based on rather simple mechanisms; essentially it is a tailored combination of delta coding and context-based modeling. Following the methodology of the preceding work presenting Polycomp (Tomasi, 2016), we compress (*i*) the ephemeris table of Ganymede, and (*ii*) the publicly available timelines recorded by LFI, an array of microwave radiometers aboard the ESA Planck spacecraft. In the former case (with small data) the compression ratio advantage of DeltaComp over Polycomp is by a factor of about 1.4. For the Planck data, of size 4.24TB, the archives produced by DeltaComp are almost six times smaller than those from Polycomp, at somewhat smaller median quantization error (and much smaller maximum error), which translates to only 66GB of required storage. Importantly, DeltaComp is over three orders of magnitude faster than Polycomp.

1. Introduction

Astronomy is one of several fields (including also bioinformatics and high energy physics) in which modern instruments produce huge volumes of data (Stephens et al., 2015), for example, the Australian Square Kilometre Array Pathfinder (ASKAP) project acquires several terabytes of sample image data per second and upon completion it is expected to require storage of size about 1 exabyte (= 10^{18} bytes) per year. Merely storing those data is challenging and demands appropriately specialized data compression techniques, formats and related tools. Representing data succinctly is beneficial not only for storage and transmission costs, but may also speed up computations (analyses) by reducing I/O processing.

In astronomy, it is quite typical to deal with tabular numeric data, with strong correlation in columns. Recently, Tomasi (2016) presented Polycomp, a configurable compression tool to handle astronomical timelines. For example, one of the experiments presented in his paper was to compress (a portion of) the recently released Planck timelines (Planck Collaboration ES, 2015), limited to the timelines of the Low Frequency Instrument (LFI), which require over 4TB of disk space. His algorithm, described in detail in Section 3, applies differential encoding to the input data, then approximates them through polynomials of some (possibly low) order, and at the end applies a compressor from an existing library (zlib or bzip2). This is a lossy scheme, involving quantization, which means that the original input can only be approximately recovered during the decompression. The maximum error per input sample is a program parameter (with a clear tradeoff between the maximum error and the attained compression ratio).

In this work, we simplify and improve the Polycomp approach. We do not (explicitly) apply polynomial approximation, but use a higher-order differential coding. Our backend compression is also more appropriate to the obtained output of the transform. As a result, our algorithm, DeltaComp, achieves a compression ratio about an order of magnitude higher than Polycomp on, e.g., the timelines of the LFI instrument onboard the Planck spacecraft. The compression reduces the storage from 4.24TB, in the uncompressed form, to about 66GB, for a reasonable lossy setting. In speed, the difference is even more striking in favour of DeltaComp.

2. Data compression basics

We start with a few definitions. A *compression algorithm* takes an input $\{d_i\}$ of n symbols, of b bits each, and compresses them to mb bits. In practice, b is often 8, which corresponds to bytes (letters of text, pixels of greyscale images etc. are often kept in single bytes) or 32 (32-bit integers, single-precision floating-point numbers), or 64 (double-precision floating-point numbers). The *compression ratio* is defined as $C_r = n/m$. If C_r is very close to 1, we say that the input data are incompressible (at least with the applied algorithm). In many domains, including measurement data in astronomy, we are satisfied with *lossy*

* Corresponding author.
E-mail address: sebastian.deorowicz@polsl.pl (S. Deorowicz).

https://doi.org/10.1016/j.newast.2018.06.006
Received 26 February 2018; Received in revised form 8 May 2018; Accepted 12 June 2018
Available online 15 June 2018
1384-1076/

original data	d0	d1	d2	d3	d4	d5	bytes for d3
1.67570799242307356	34910583	34910583	34910583	34910583	34910583	34910583	{246, 2, 20, 177, 119}
1.67760573897480803	34950120	39537	-34871046	-69781629	-104692212	-139602795	{247, 4, 40, 200, 125}
1.67950255680698390	34989637	39517	-20	34871026	104652655	209344867	{246, 2, 20, 22, 242}
1.68139843250048204	35029134	39497	-20	0	-34871026	-139523681	{0}
1.68329335261744162	35068612	39478	-19	1	1	34871027	{2}
1.68518730370098435	35108069	39457	-21	-2	-3	-4	{3}
1.68708027227493873	35147506	39437	-20	1	3	6	{2}
1.68897224484356157	35186922	39416	-21	-1	-2	-5	{1}
1.69086320789126776	35226317	39395	-21	0	1	3	{0}
1.69275314788235054	35265691	39374	-21	0	0	-1	{0}
1.69464205126071166	35305043	39352	-22	-1	-1	-1	{1}
1.69652990444958496	35344373	39330	-22	0	1	2	{0}
1.69841669385126504	35383681	39308	-22	0	0	-1	{0}
1.70030240584683567	35422967	39286	-22	0	0	0	{0}
1.70218702679589717	35462230	39263	-23	-1	-1	-1	{1}
1.70407054303629546	35501470	39240	-23	0	1	2	{0}
1.70595294088385296	35540686	39216	-24	-1	-1	-2	{1}
1.70783420663209706	35579879	39193	-23	1	2	3	{2}
1.70971432655199407	35619048	39169	-24	-1	-2	-4	{1}
1.71159328689167856	35658193	39145	-24	0	1	3	{0}
1.71347107387618691	35697314	39121	-24	0	0	-1	{0}
1.71534767370719066	35736410	39096	-25	-1	-1	-1	{1}
1.71722307256272999	35775481	39071	-25	0	1	2	{0}
1.71909725659694868	35814526	39045	-26	-1	-1	-2	{1}
1.72097021193983091	35853546	39020	-25	1	2	3	{2}
1.72284192469693442	35892540	38994	-26	-1	-2	-4	{1}
1.72471238094913093	35931508	38968	-26	0	1	3	{0}
1.72658156675234209	35970449	38941	-27	-1	-1	-2	{1}
1.72844946813728040	36009364	38915	-26	1	2	3	{2}
1.73031607110918406	36048251	38887	-28	-2	-3	-5	{3}

Fig. 1. Delta coding on a small sample of quantized LFI27M data (Θ angles). Five rounds of delta coding shown. The most successful number of rounds is 3 and for the respective obtained values our byte coding variant is applied (the rightmost column).

Table 1
Ganymede compression ratios and compression times (in seconds), per coordinate. Polycomp worked with 48 threads, DeltaComp was single-threaded.

Algorithm	Ratio			Compression time		
	x	y	z	x	y	z
Polycomp	13.15	13.26	24.35	93.21	93.46	35.80
DeltaComp	17.89	18.15	35.70	0.30	0.33	0.27

compression, which means that the decompressed (recovered) data may only approximate the input submitted to the compression procedure. Many error measures are used in lossy compression (e.g., for multimedia data), yet here, following Tomasi (2016), we characterize the quality of the approximation by the maximal error $\epsilon_c = \max_{i=1 \ldots n}|d_i - \widetilde{d_i}|$, where $\widetilde{d_i}$ are the decompressed symbols.

Data compression techniques consist of modeling and coding. The modeling phase is the way to look at the input data: they may be perceived as bits, bytes, pixels, words, 64-bit floating-point numbers, and so on. They may be transformed in different ways, in order to seek for various kinds of repetitions and regularities. The output of the modeling phase is submitted to a coder, which basically works according to the golden rule of compression: shorter codewords should be assigned to frequent (i.e., more probable) symbols, and longer codewords to the symbols which are rare.

Let us now briefly present some compression techniques. They are essentially lossless, but at least some of them are often used, as components, in lossy solutions. More information on the presented ideas can be found, e.g., in Salomon and Motta (2010).

2.1. Run-length encoding (RLE)

RLE is probably the simplest and most obvious compression technique, with very limited applications. It replaces runs of the same

Table 2
Planck compression ratios and times. Times are given in format h:mm. Polycomp time estimated for 24-core Xeon machine. DeltaComp times for parallel execution of the algorithm for various datasets using 24 cores.

Algorithm	Max. error	Med. error	Compression ratios			Compression times		
	[marcsec]	[marcsec]	30 GHz	44 GHz	70 GHz	30 GHz	44 GHz	70 GHz
Polycomp*	1000	~ 30	7.39	9.18	12.67	4340	7360	31060
DeltaComp	500	~ 250	76.61	91.02	121.95	1:36	2:57	8:32
DeltaComp	50	~ 25	44.05	53.69	73.40	1:36	2:57	7:42
DeltaComp	5	~ 2.5	23.17	29.47	41.48	2:24	4:08	8:39
DeltaComp	0.5	~ 0.25	12.58	15.96	22.39	3:00	5:26	11:58